Towards Building the OceanStore Web Cache

Patrick R. Eaton
University of California, Berkeley
eaton@cs.berkeley.edu

June 10, 2002

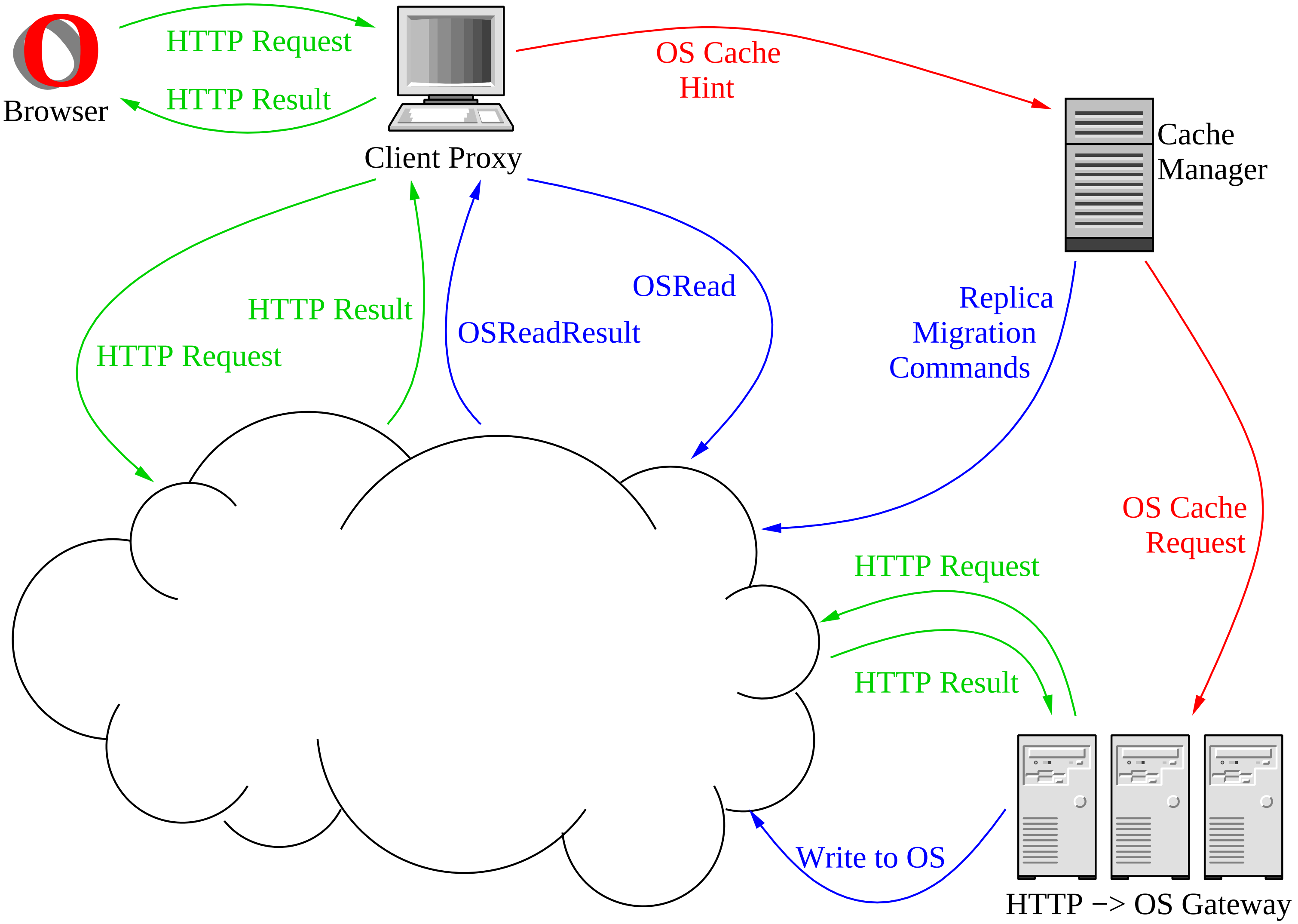
Motivation

- Traditional hierarchical web caching architectures require much maintenance and human configuration.
- We have developed a web cache architecture which exploits the features of OceanStore to be self-configuring/managing/maintaining.
  - uses Tapestry to allow cache nodes to enter and leave the network without impacting other caches
  - uses Tapestry to locate objects in the network without explicit knowledge of other caches
  - uses excess resources in the network to cache more content
- What is the cost in performance of this new architecture?

Components of the OceanStore Web Caching Architecture

- Client proxy.
  - translates a user’s web requests to check the OceanStore web cache
  - runs on same machine as user’s web browser
- HTTP to OceanStore gateway.
  - convert web content into OceanStore documents
  - hosted by regional cache provider
- Cache managers.
  - work greedily to provide best level of service to clients in the local area
  - run locally by department or organization

The OceanStore Web Cache Architecture



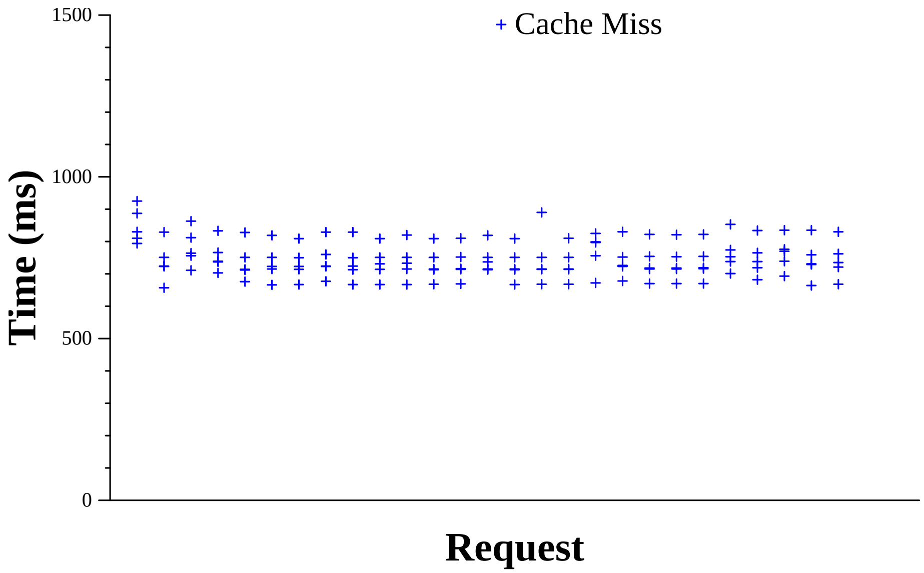
Scalability and Maintainability

- Tapestry allows nodes to enter and leave the network without notice.
- Tapestry allows us to locate service providers.
- No hierarchy or group configuration/maintenance.
- Efficient use of excess resources in the network.
- No network “hot-spots”.
- Greater aggregate read bandwidth.

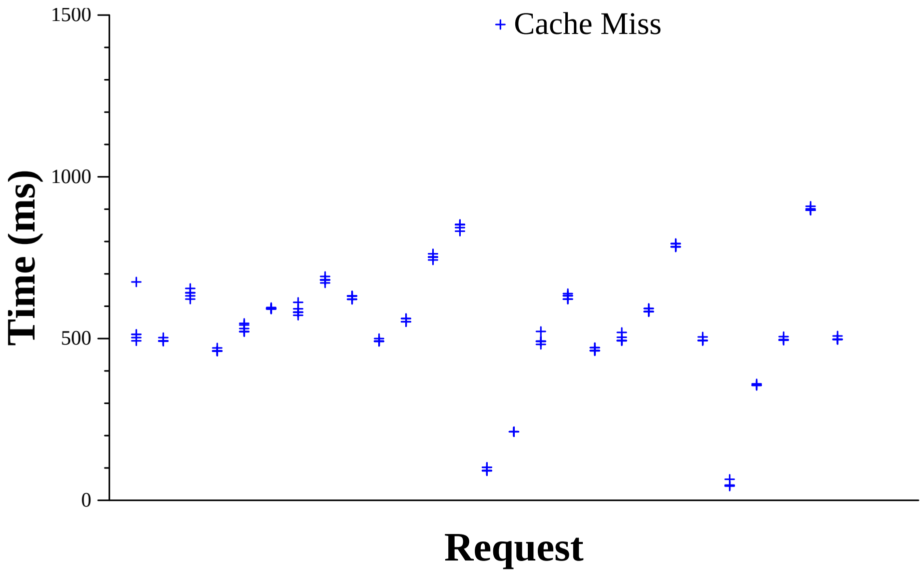
Cache Latency

- Measure the latency of a single request.
- Cache miss.
  - document is not cached on any node
  - retrieve document from origin server after lookup fails
- Local hit.
  - document is cached locally
  - can return document immediately
- Remote hit.
  - document is not cached locally but is cached on some node
  - must find node with content cached and retrieve document
- Key difference between caches.
  - OceanStore searches other caches through a series of serial Tapestry hops
  - ICP searches other caches through a parallel multicast

Cache Latency: Cache Miss



Latency of ICP Cache

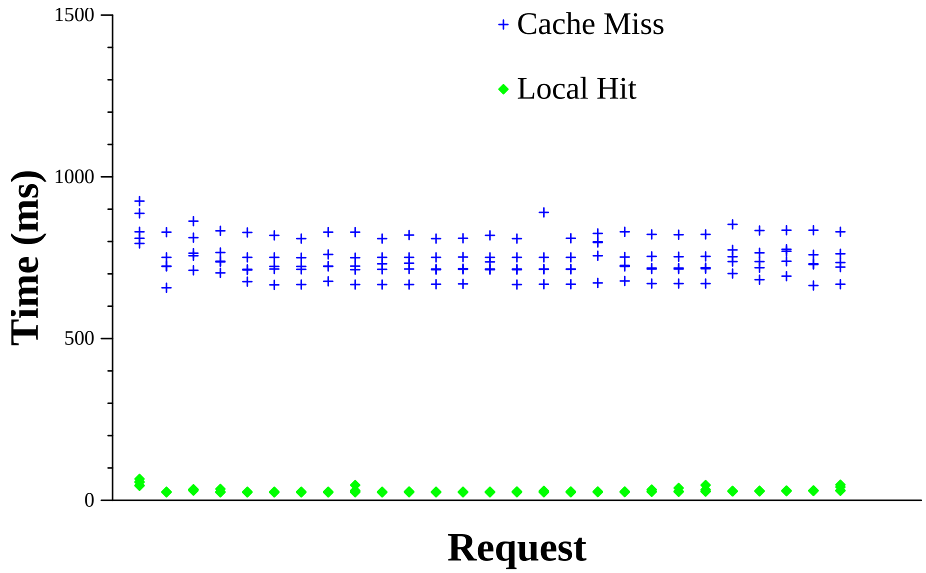


Latency of OceanStore Web Cache

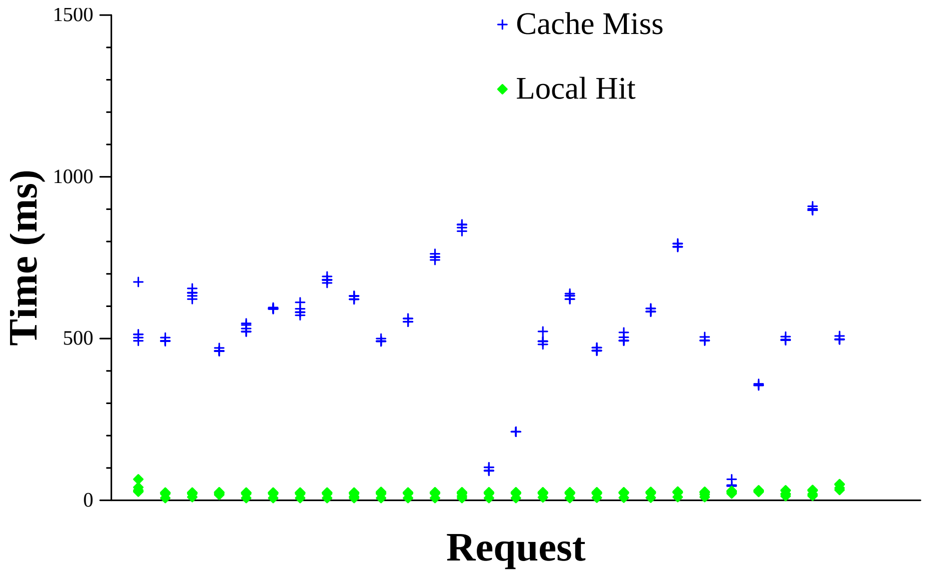
- ICP cache waits to receive all nacks before requesting the document from the origin server.

- OceanStore cache requests document from origin server when Tapestry resolves that the document is not published in the network.

Cache Latency: Local Hit



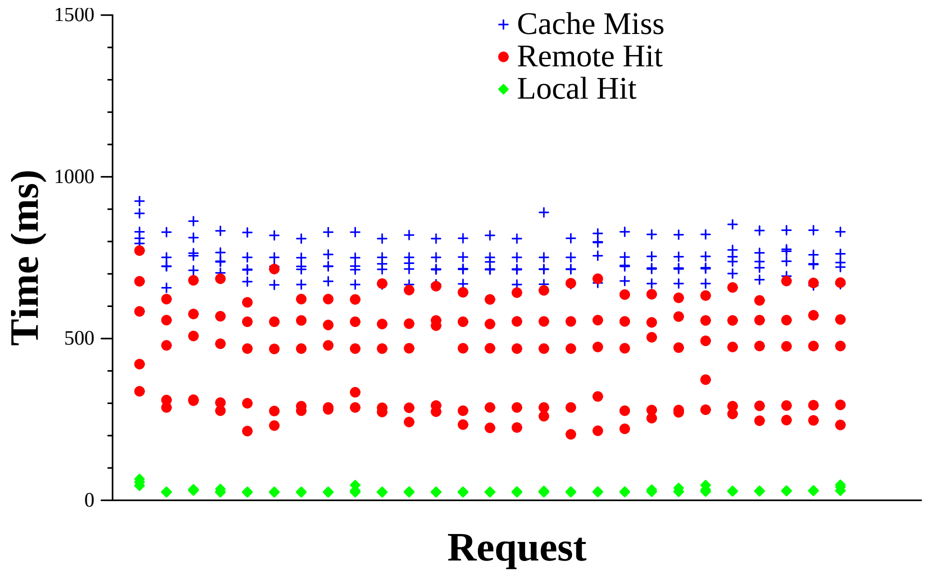
Latency of ICP Cache



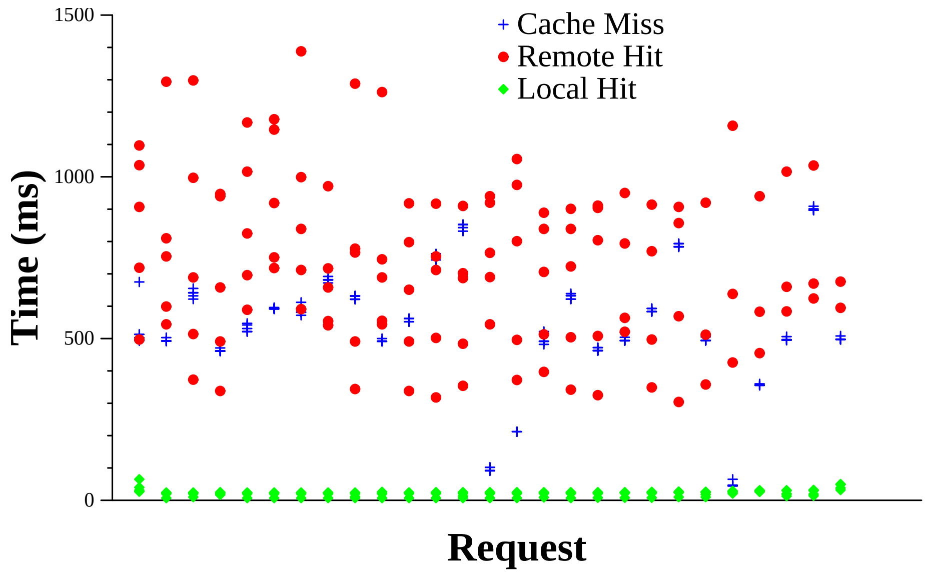
Latency of OceanStore Web Cache

- Both caches respond very quickly when document is cached locally.
- OceanStore cache actually serves close content twice as fast as the ICP cache (20 ms versus 35 ms).
  - OceanStore cache can move content to the requesting client
  - ICP cache can only move content to the proxy of the requesting client

Cache Latency: Remote Hit - The Bad News



Latency of ICP Cache

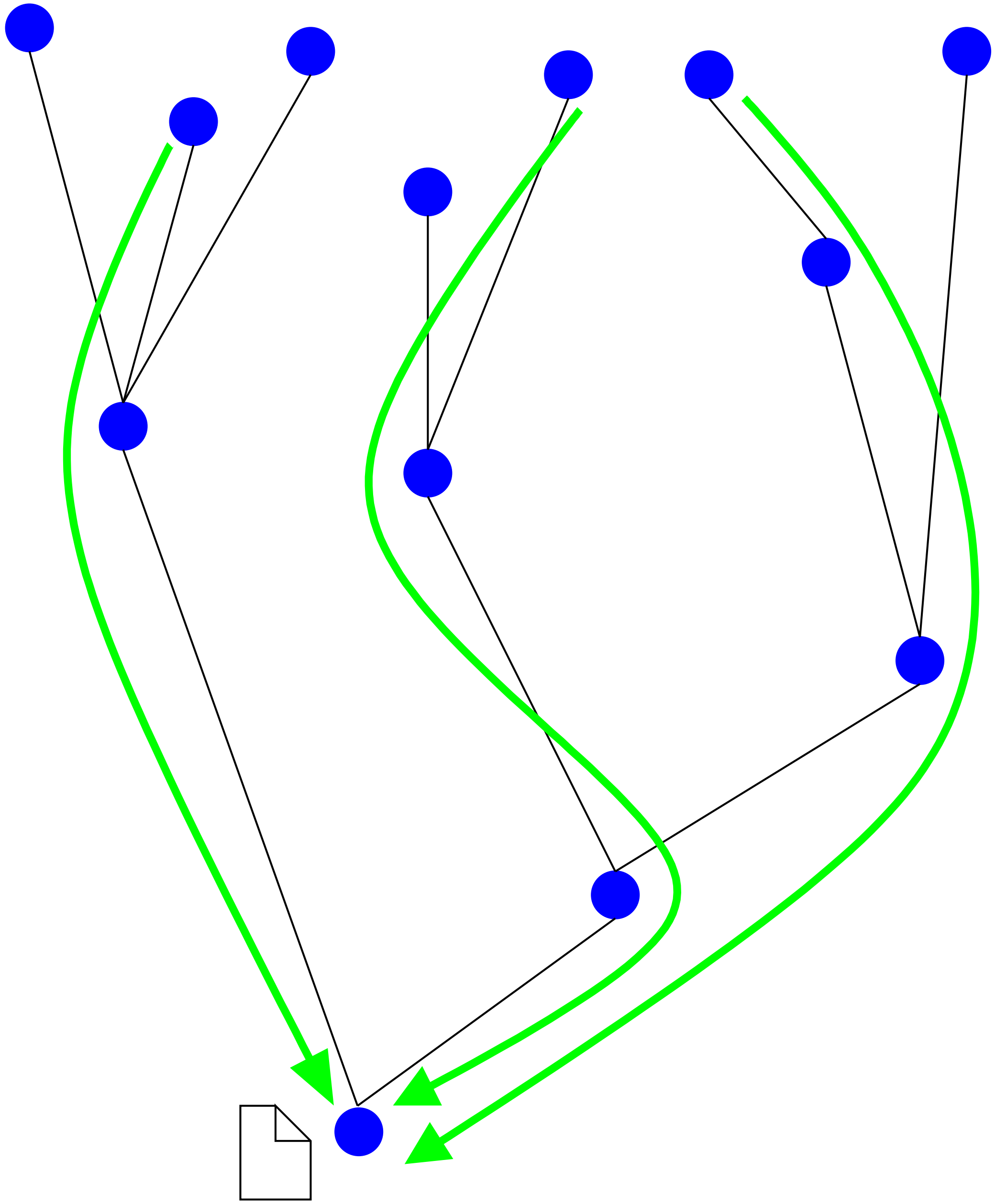


Latency of OceanStore Web Cache

- Can observe the effect of Tapestry’s hop-by-hop routing.
  - highlights the importance of managing replicas to ensure content is close to consumers
- OceanStore cache can actually serve content faster when it is nearby.

Inspiration for Replica Placement Strategy

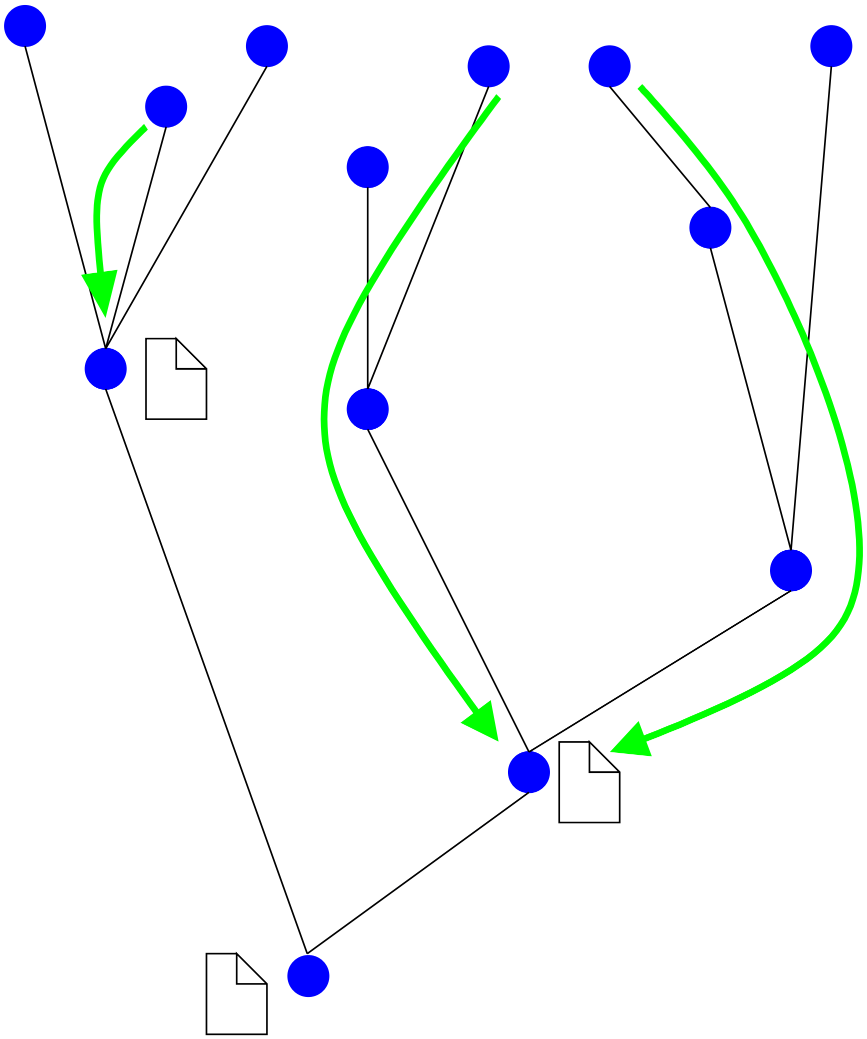
In Tapestry, object location paths combine at Tapestry nodes. Location requests are routed from the edges of the network toward the object's Tapestry root.



Replica Placement Strategy

- Idea: Place replicas at the "confluence" of location paths.

- All clients "upstream" of the replica will benefit from it.



Conclusions and Ongoing Work

- The performance of individual components is adequate.
- The key to good aggregate performance is effective replica management.
- Ongoing work:
  - Implement replica management in the cache managers.
  - Explore use of Tapestry “time-outs” to reduce the cost of remote hits.
  - Measure the effect of using idle resources in the network.
  - Find appropriate workloads/load generators for measuring the system.